



Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №3
із дисципліни «Технології розроблення програмного забезпечення»
Тема: «Основи проектування розгортання»
Тема проекту: «Музичний програвач»

Виконала:
студентка групи ІА-31:
Мартищенко І.С.

Перевірив:
Мягкий М.Ю.

Тема: Основи проектування розгортання.

Мета: Навчитися проектувати діаграми розгортання та компонентів для системи

що проектується, а також розробляти діаграми взаємодії, а саме діаграми послідовностей, на основі сценаріїв зроблених в попередній лабораторній роботі.

Завдання

- Ознайомитись з короткими теоретичними відомостями.
- Проаналізувати діаграми створені в попередній лабораторній роботі а також тему системи та спроектувати діаграму розгортання використання відповідно до обраної теми лабораторного циклу.
- Розробити діаграму компонентів для проектованої системи.
- Розробити діаграму розгортання для проектованої системи.
- Розробити як мінімум дві діаграми послідовностей для сценаріїв прописаних в попередній лабораторній роботі.
- На основі спроектованих діаграм розгортання та компонентів доопрацювати програмну частину системи. Реалізація системи, додатково до попередньої реалізації, повинна містити як мінімум дві візуальні форми. В системі вже повинен бути повністю реалізована архітектура (повний цикл роботи з даними від вводу на формі до збереження їх в БД і подальшій виборці з БД та відображенням на UI).
- Підготувати звіт щодо виконання лабораторної роботи. Поданий звіт повинен містити: діаграму розгортання з описом, діаграму компонентів системи з описом, діаграми послідовностей, а також вихідний код системи, який було додано в цій лабораторній роботі.

Хід роботи

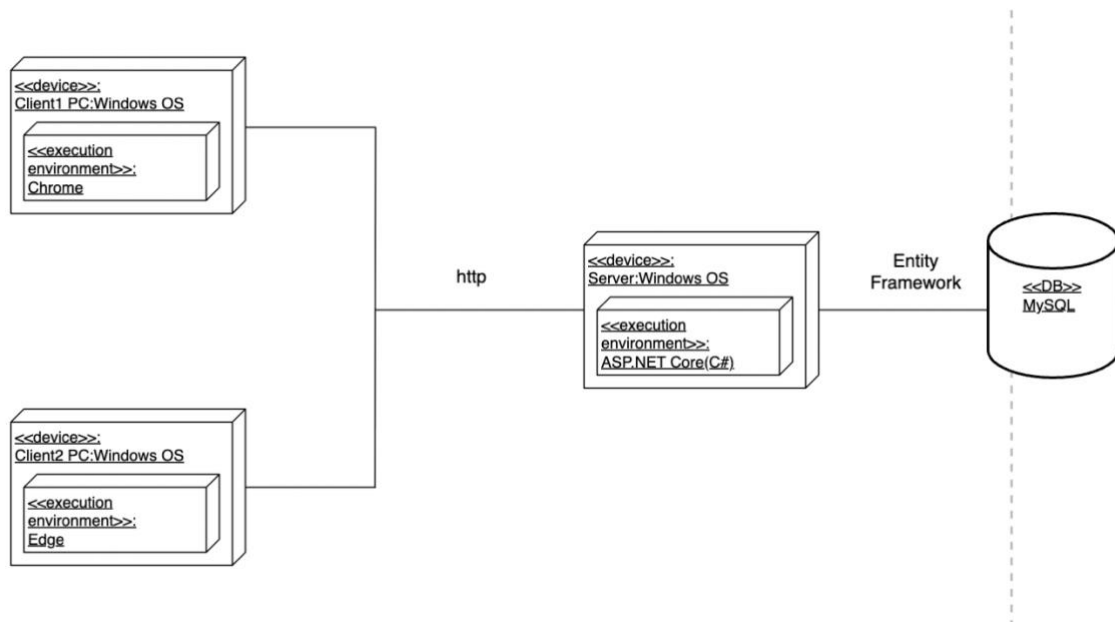


Рис-1 Діаграма розгортання

На діаграмі розгортання зображено інфраструктуру вебзастосунку «Музичний програвач», у якій клієнти працюють на операційній системі Windows та взаємодіють із системою через браузери Chrome і Edge.

Вебзастосунок розгорнутий на сервері під керуванням Windows OS і виконується у середовищі ASP.NET Core (C#).

Для доступу до бази даних MySQL застосунок використовує ORM-технологію Entity Framework Core, що забезпечує зручну взаємодію між серверною частиною та базою даних.

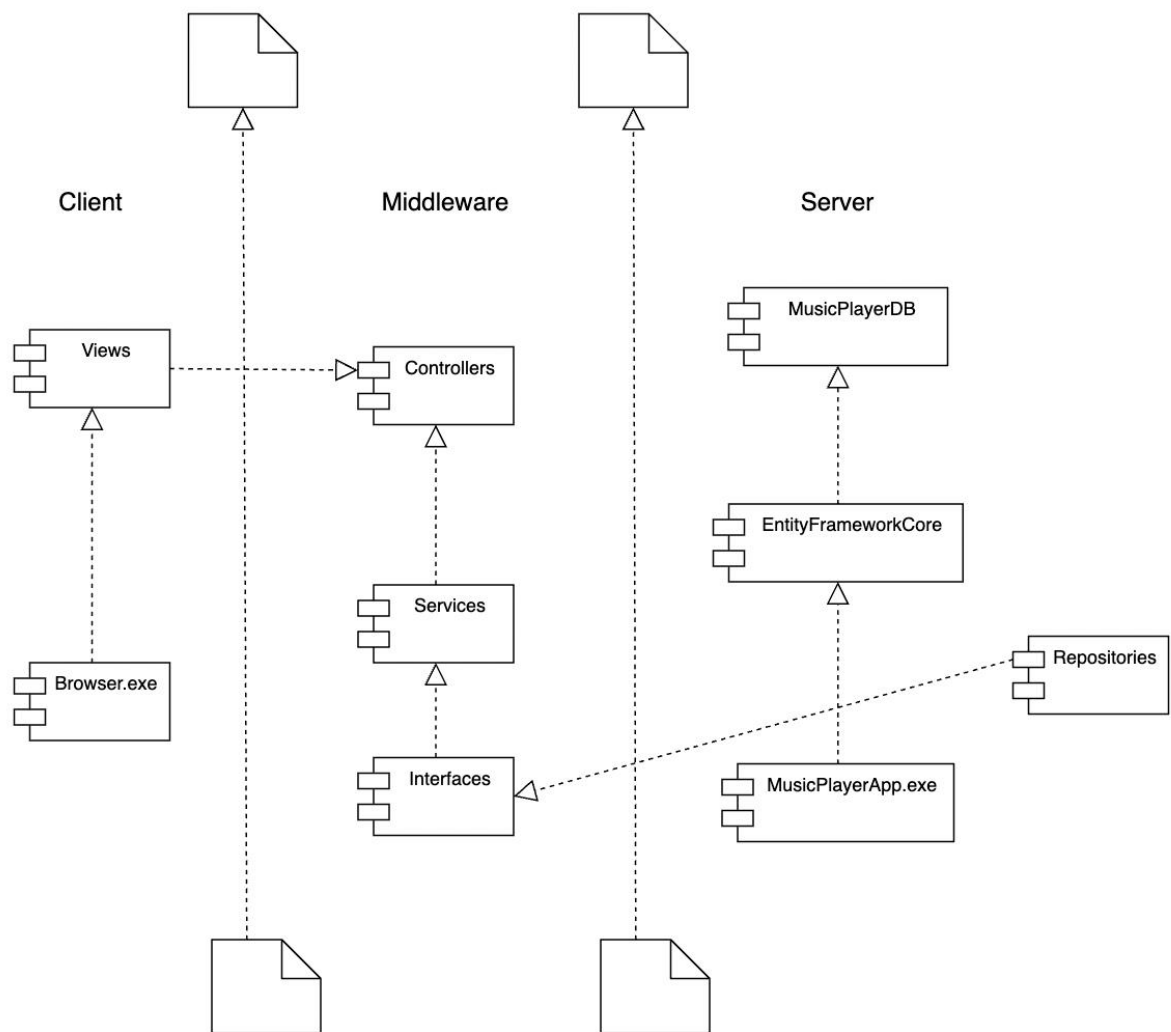


Рис-2 Діаграма компонентів для проєктованої системи

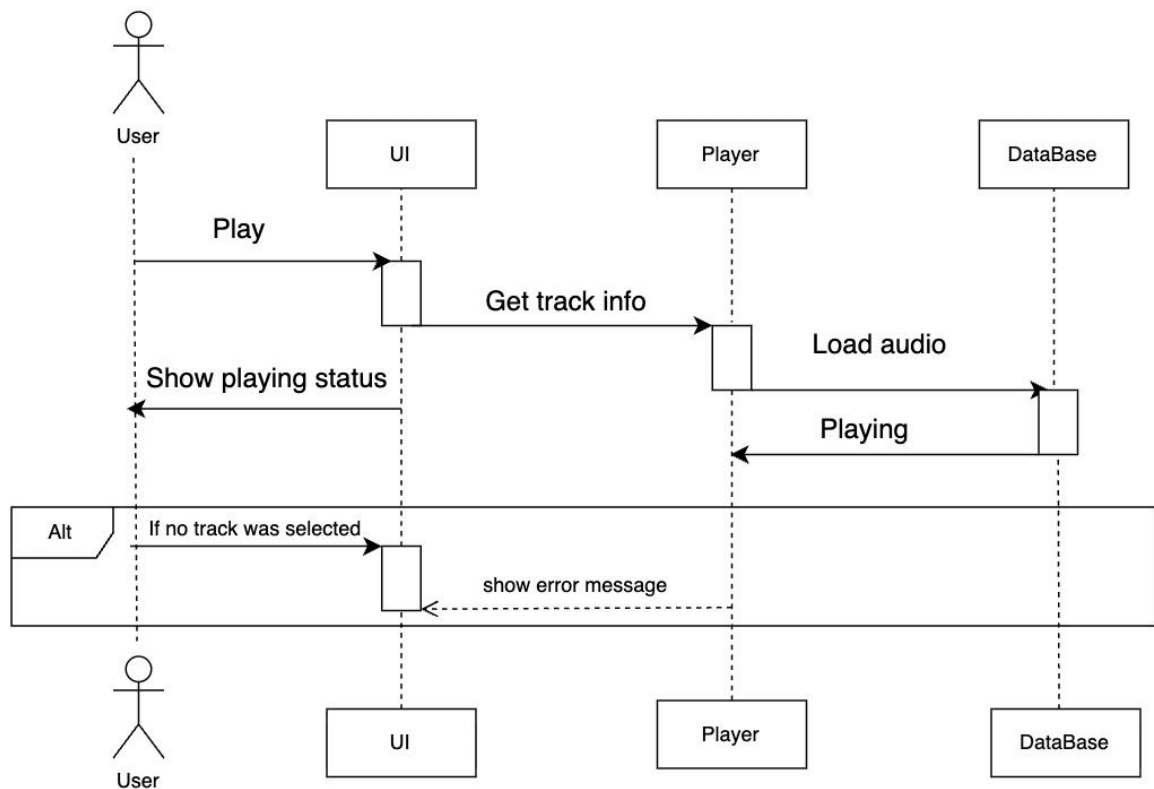


Рис-3 Діаграма послідовностей(сценарій 1)

Сценарій 1:

На діаграмі послідовності зображено процес відтворення музичного треку користувачем у системі «MusicPlayer».

Користувач ініціює процес, натискаючи кнопку «Відтворити» у графічному інтерфейсі (UI). Система через інтерфейс отримує інформацію про обраний аудіофайл і передає її модулю Player для подальшої обробки.

Модуль Player звертається до бази даних (DataBase) для отримання шляху до аудіофайлу та завантаження необхідних даних. Після цього плеєр починає відтворення треку, а інтерфейс користувача оновлює стан - відображаючи, що відтворення триває.

У альтернативному потоці (Alt) передбачено обробку виняткових ситуацій: якщо користувач не обрав жодного треку, система показує повідомлення про помилку із закликом спочатку вибрати файл.

Передумови: користувач відкрив програму та обрав аудіофайл.

Постумови: система починає відтворення обраного треку.

Основні учасники:

User - ініціює відтворення;

UI - відображає інтерфейс та обробляє дії користувача;

Player - відповідає за відтворення аудіо;

DataBase - зберігає інформацію про треки (шлях до файлу, назву тощо).

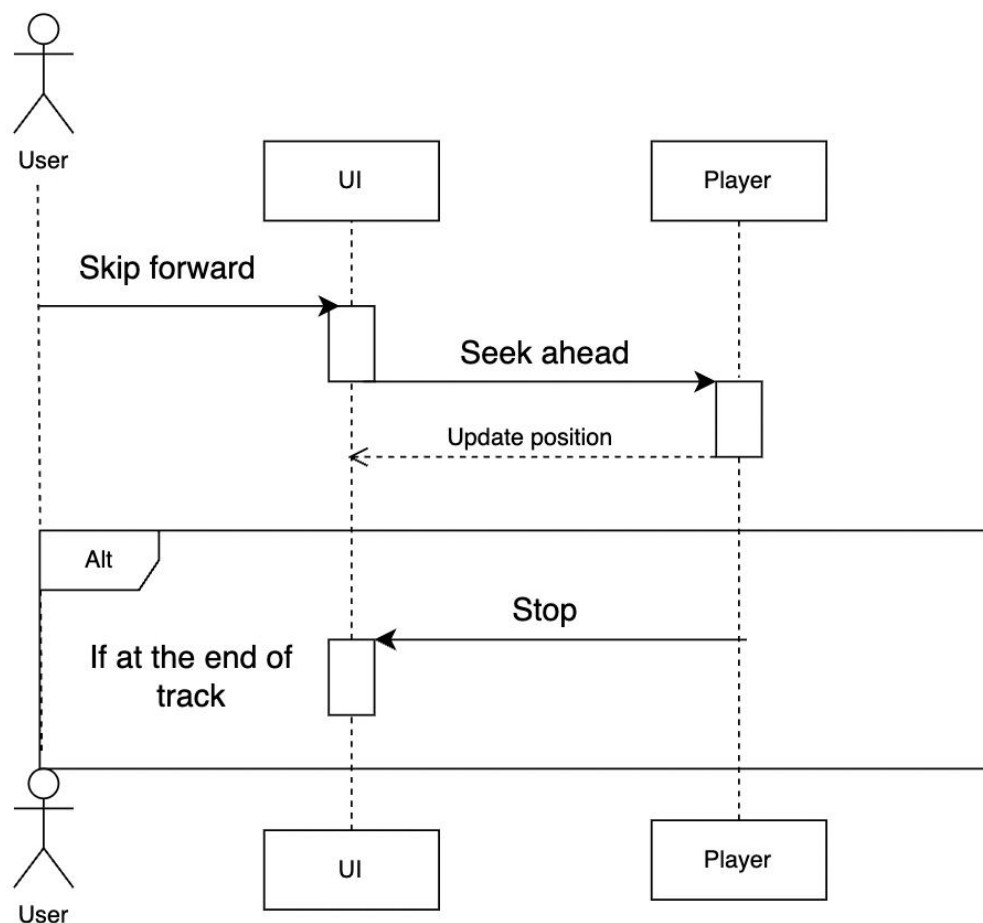


Рис-4 Діаграма послідовностей(сценарій 2)

Сценарій 2:

На діаграмі послідовності показано процес перемотування треку вперед під час його відтворення.

Користувач, який уже слухає трек, натискає кнопку «Перемотати вперед» у графічному інтерфейсі (UI).

Інтерфейс передає команду модулю Player, який змінює поточну позицію у відтворюваному аудіофайлі, перемотуючи трек на кілька секунд уперед.

Після зміни позиції Player оновлює дані про поточну точку відтворення та повідомляє UI для оновлення відображення часу треку.

Відтворення продовжується з нової позиції у файлі.

В альтернативному потоці (Alt) розглядається випадок, коли перемотування перевищує довжину треку - у цьому разі Player зупиняє відтворення, і UI відображає стан зупинки.

Передумови: трек уже відтворюється, інтерфейс активний.

Постумови: відтворення продовжується з нової позиції в аудіофайлі.

Взаємодіючі сторони: користувач, інтерфейс користувача (UI), плеєр (Player).

Примітки: перемотування доступне лише під час активного відтворення треку.

Реєстрація

Ім'я користувача

Пароль

Зареєструватися

Вже маєш акаунт? [Увійти](#)

Рис-5 Форма реєстрації



 Id	Username	PasswordHash	
16898e8e-9efb-47da-906a-d1877b6af60b	Ірина	5994471ABB01112AFCC18159F6CC74B4F511B99806DA59B3CAF5A9C173CACFC5	

Рис-6 Користувач в базі даних






Вийти з акаунтуВийти з програми

Мій плейлист

Назва треку (необов'язково)

Відкрити аудіофайл
Choose fileNo file chosenДодати трек

0:02 / 3:52

 -10с +10с

Зараз грає: 14adadca9148f2d77303bbe3c4107435.mp3

Треки

Назва	Дія
14adadca9148f2d77303bbe3c4107435.mp3	<div>ВідтворитиВидалити</div>

Рис-7 Музичний програвач

Відкрити аудіофайл

Choose file14adadca91...107435.mp3Додати трек

Рис-8 Додавання треку

Назва	Дія
14adadca9148f2d77303bbe3c4107435.mp3	<div>ВідтворитиВидалити</div>

Рис-9 Доданий трек


 Id	Title	AddedAt	FileName	OwnerUsername	RelativeUrl
ca49f122-548e-443d-8fe1-66ba997b582c	Shape of you	2025-10-21 21:38:11	5817017e-ab5a-40b0-8e5e-220cf415fca4.mp3	Ірина	/uploads/5817017e-ab5a-40b0-8e5e-220cf415fca4.mp3

Рис-10 Трек в базі даних


```

~/MusicPlayerWeb/Controllers .ms;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Mvc;
using MusicPlayerWeb.Services;

namespace MusicPlayerWeb.Controllers;

1 reference
public class AccountController : Controller
{
    3 references
    private readonly IUserService _users;

    0 references
    public AccountController(IUserService users) => _users = users;

    [HttpGet]
    2 references
    public IActionResult Login(string? returnUrl = null)
    {
        ViewBag.ReturnUrl = returnUrl;
        return View();
    }

    [HttpPost]
    2 references
    public async Task<IActionResult> Login(string username, string password, string? returnUrl = null)
    {
        if (_users.ValidateCredentials(username, password))
        {
            var claims = new List<Claim> { new Claim(ClaimTypes.Name, username) };
            var identity = new ClaimsIdentity(claims, CookieAuthenticationDefaults.AuthenticationScheme);
            var principal = new ClaimsPrincipal(identity);
            await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, principal);
            return Redirect(returnUrl ?? "/");
        }

        ViewBag.Error = "Невірне ім'я користувача або пароль.";
        return View();
    }

    [HttpGet]
    0 references
    public IActionResult Register() => View();

    [HttpPost]
    0 references
    public IActionResult Register(string username, string password)
    {
        if (_users.Register(username, password, out var err))
            return RedirectToAction(nameof(Login));

        ViewBag.Error = err;
        return View();
    }

    0 references
    public async Task<IActionResult> Logout()
    {
        await HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);
        return RedirectToAction(nameof(Login));
    }
}

```

Код-1 Controllers-AccountController.cs

rollers / HomeController.cs / ...

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using MusicPlayerWeb.Services;

namespace MusicPlayerWeb.Controllers;

[Authorize]
1 reference
public class HomeController : Controller
{
    2 references
    private readonly ITrackService _tracks;

    0 references
    public HomeController(ITrackService tracks) => _tracks = tracks;

    0 references
    public IActionResult Index()
    {
        var username = User.Identity!.Name!;
        var model = _tracks.GetForUser(username);
        return View(model);
    }
}
```

Код-2 Controllers-HomeController.cs

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Hosting;

namespace MusicPlayerWeb.Controllers;

[Authorize]
1 reference
public class SystemController : Controller
{
    2 references
    private readonly IHostApplicationLifetime _lifetime;

    0 references
    public SystemController(IHostApplicationLifetime lifetime) => _lifetime = lifetime;

    [HttpPost]
    0 references
    public IActionResult Exit()
    {
        _lifetime.StopApplication();
        return Content("Завершения работи...");
    }
}
```

Код-3 Controllers-SystemController.cs

```

Controllers / TracksController.cs / ...
1  using Microsoft.AspNetCore.Authorization;
2  using Microsoft.AspNetCore.Mvc;
3  using MusicPlayerWeb.Services;
4
5  namespace MusicPlayerWeb.Controllers;
6
7  [Authorize]
1 reference
8  public class TracksController : Controller
9  {
2 references
10     private readonly IWebHostEnvironment _env;
3 references
11     private readonly ITrackService _tracks;
12
0 references
13     public TracksController(IWebHostEnvironment env, ITrackService tracks)
14     {
15         _env = env;
16         _tracks = tracks;
17     }
18
19     [HttpPost]
0 references
20     public async Task<ActionResult> Add(string? title, IFormFile file)
21     {
22         if (file == null || file.Length == 0)
23         {
24             TempData["Error"] = "Будь ласка, вибери аудіофайл.";
25             return RedirectToAction("Index", "Home");
26         }
27
28         var uploads = Path.Combine(_env.WebRootPath, "uploads");
29         Directory.CreateDirectory(uploads);
30
31         var safeName = $"{Guid.NewGuid()}{Path.GetExtension(file.FileName)}";
32         var path = Path.Combine(uploads, safeName);
33         await using (var stream = System.IO.File.Create(path))
34         {
35             await file.CopyToAsync(stream);
36         }
37         var rel = $"/uploads/{safeName}";
38         _tracks.AddForUser(User.Identity!.Name!, title ?? file.FileName, safeName, rel);
39
40         return RedirectToAction("Index", "Home");
41     }
42
43     [HttpPost]
0 references
44     public IActionResult Delete(Guid id)
45     {
46         var username = User.Identity!.Name!;
47         if (_tracks.Delete(username, id))
48         {
49             return RedirectToAction("Index", "Home");
50         }
51     }
52 }
53

```

Код-4 Controllers-TrackController.cs

```

1 using Microsoft.EntityFrameworkCore;
2 using MusicPlayerWeb.Models;
3
4 namespace MusicPlayerWeb.Data
5 {
6     10 references
7     public class AppDbContext : DbContext
8     {
9         3 references
10        public DbSet<User> Users => Set<User>();
11        4 references
12        public DbSet<Track> Tracks => Set<Track>();
13
14        0 references
15        public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }
16
17        0 references
18        protected override void OnModelCreating(ModelBuilder b)
19        {
20            // Users
21            b.Entity<User>(e =>
22            {
23                e.HasKey(x => x.Id);
24                e.Property(x => x.Username).IsRequired().HasMaxLength(100);
25                e.Property(x => x.PasswordHash).IsRequired();
26                e.HasIndex(x => x.Username).IsUnique();
27            });
28
29            // Tracks
30            b.Entity<Track>(e =>
31            {
32                e.HasKey(x => x.Id);
33                e.Property(x => x.Title).IsRequired().HasMaxLength(256);
34                e.Property(x => x.FileName).IsRequired().HasMaxLength(256);
35                e.Property(x => x.RelativeUrl).IsRequired().HasMaxLength(512);
36                e.Property(x => x.OwnerUsername).HasMaxLength(100);
37                e.HasIndex(x => new { x.OwnerUsername, x.AddedAt });
38            });
39        }
40    }
41 }

```

Код-5 Data-AppDbcontext.cs

```

1 namespace MusicPlayerWeb.Models;
2
3 11 references
4 public class Track
5 {
6     3 references
7     public Guid Id { get; set; } = Guid.NewGuid();
8     4 references
9     public string Title { get; set; } = default!;
10    2 references
11    public string FileName { get; set; } = default!;
12    3 references
13    public string RelativeUrl { get; set; } = default!;
14    3 references
15    public DateTime AddedAt { get; set; } = DateTime.UtcNow;
16    5 references
17    public string? OwnerUsername { get; set; }
18 }

```

Код-6 Models-Track.cs

```

1 namespace MusicPlayerWeb.Models;
2
3 public class User
4 {
5     public Guid Id { get; set; } = Guid.NewGuid();
6     public string Username { get; set; } = default!;
7     public string PasswordHash { get; set; } = default!;
8 }

```

Код-7 Models-User.cs

```

{
  "profiles": {
    "MusicPlayerWeb": {
      "commandName": "Project",
      "dotnetRunMessages": true,
      "launchBrowser": true,
      "launchUrl": "https://localhost:5001/",
      "applicationUrl": "https://localhost:5001;http://localhost:5000",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    }
  }
}

```

Код-8 Properties-launchSettings.json

```

1  using Microsoft.EntityFrameworkCore;
2  using MusicPlayerWeb.Data;
3  using MusicPlayerWeb.Models;
4
5  namespace MusicPlayerWeb.Services;
6
7  2 references
8  public class DbTrackService : ITrackService
9  {
10     7 references
11     private readonly AppDbContext _db;
12
13     0 references
14     public DbTrackService(AppDbContext db) => _db = db;
15
16     2 references
17     public IEnumerable<Track> GetForUser(string username)
18     {
19         return _db.Tracks
20             .AsNoTracking()
21             .Where(t => t.OwnerUsername == username)
22             .OrderByDescending(t => t.AddedAt)
23             .ToList();
24     }
25
26     2 references
27     public Track AddForUser(string username, string title, string fileName, string relUrl)
28     {
29         var track = new Track
30         {
31             Title = string.IsNullOrEmpty(title) ? fileName : title.Trim(),
32             FileName = fileName,
33             RelativeUrl = relUrl,
34             OwnerUsername = username,
35             AddedAt = DateTime.UtcNow
36         };
37         _db.Tracks.Add(track);
38         _db.SaveChanges();
39         return track;
40     }
41
42     2 references
43     public bool Delete(string username, Guid id)
44     {
45         var tr = _db.Tracks.FirstOrDefault(t => t.Id == id && t.OwnerUsername == username);
46         if (tr == null) return false;
47
48         _db.Tracks.Remove(tr);
49         _db.SaveChanges();
50         return true;
51     }
52 }

```

Код-9 Services-DbTrackService.cs

```

vices > C:\DbUserService.cs > ...
1  ~/MusicPlayerWeb/Services cryptography;
2  using System.Text;
3  using Microsoft.EntityFrameworkCore;
4  using MusicPlayerWeb.Data;
5  using MusicPlayerWeb.Models;
6
7  namespace MusicPlayerWeb.Services;
8
9  2 references
10 public class DbUserService : IUserService
11 {
12     5 references
13     private readonly AppDbContext _db;
14
15     0 references
16     public DbUserService(AppDbContext db) => _db = db;
17
18     2 references
19     public bool Register(string username, string password, out string error)
20     {
21         error = "";
22
23         if (string.IsNullOrEmpty(username) || string.IsNullOrEmpty(password))
24         {
25             error = "Ім'я користувача і пароль обов'язкові.";
26             return false;
27         }
28
29         username = username.Trim();
30
31         if (_db.Users.Any(u => u.Username == username))
32         {
33             error = "Такий користувач вже існує.";
34             return false;
35         }
36
37         var user = new User
38         {
39             Username = username,
40             PasswordHash = Hash(password)
41         };
42
43         _db.Users.Add(user);
44         _db.SaveChanges();
45         return true;
46     }
47
48     2 references
49     public bool ValidateCredentials(string username, string password)
50     {
51         var hash = Hash(password);
52         return _db.Users.AsNoTracking()
53             .Any(u => u.Username == username && u.PasswordHash == hash);
54     }
55
56     2 references
57     private static string Hash(string input)
58     {
59         using var sha = SHA256.Create();
60         var bytes = sha.ComputeHash(Encoding.UTF8.GetBytes(input));
61         return Convert.ToHexString(bytes);
62     }
63 }

```

Код-9 Services-DbUserService.cs

```

rvice > ITrackService.cs > ...
1 using MusicPlayerWeb.Models;
2
3 namespace MusicPlayerWeb.Services;
4
5 6 references
6 public interface ITrackService
7 {
8     2 references
8     IEnumerable<Track> GetForUser(string username);
9     2 references
8     Track AddForUser(string username, string title, string fileName, string relUrl);
9     2 references
9     bool Delete(string username, Guid id);
0 }
1

```

Код-10 Services-ITrackService.cs

```

.es / IUserService.cs / ...
using MusicPlayerWeb.Models;

namespace MusicPlayerWeb.Services;

4 references
public interface IUserService
{
    2 references
    bool Register(string username, string password, out string error);
    2 references
    bool ValidateCredentials(string username, string password);
}

```

Код-11 Services-IUserService.cs

```

's > _ViewImports.cshtml
@using MusicPlayerWeb
@using MusicPlayerWeb.Models
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers

```

Код-12 Views-_ViewImports.cshtml

```

@{
    Layout = "_Layout";
}

```

Код-13 Views-_ViewStart.cshtml

s / Account / = Login.cshtml

```
@{
    ViewBag.Title = "Увійти";
}

<h2>Увійти</h2>
@if (ViewBag.Error != null)
{
    <div class="alert">@ViewBag.Error</div>
}
<form method="post">
    <label>Ім'я користувача</label>
    <input name="username" required />
    <label>Пароль</label>
    <input name="password" type="password" required />
    <button type="submit">Увійти</button>
</form>
<p>Немає облікового запису? <a href="/Account/Register">Зареєструватися</a></p>
```

Код-14 Views – Account – Login.cshtml

NS / Account / = Register.cshtml

```
@{
    ViewBag.Title = "Реєстрація";
}
<h2>Реєстрація</h2>
@if (ViewBag.Error != null)
{
    <div class="alert">@ViewBag.Error</div>
}
<form method="post">
    <label>Ім'я користувача</label>
    <input name="username" required />
    <label>Пароль</label>
    <input name="password" type="password" required />
    <button type="submit">Зареєструватися</button>
</form>
<p>Вже маєш акаунт? <a href="/Account/Login">Увійти</a></p>
```

Код-15 Views – Account – Register.cshtml

```

@model IEnumerable<MusicPlayerWeb.Models.Track>

<h2>Мій плейлист</h2>

@if (TempData["Error"] != null)
{
    <div class="alert">@TempData["Error"]</div>
}

<section class="uploader">
    <form asp-controller="Tracks" asp-action="Add" method="post" enctype="multipart/form-data">
        <label>Назва треку (необов'язково)</label>
        <input name="title" />
        <label>Відкрити аудіофайл</label>
        <input type="file" name="file" accept="audio/*" required />
        <button type="submit">Додати трек</button>
    </form>
</section>

<section class="player">
    <audio id="player" controls preload="metadata" style="width:100%">
        <source id="playerSource" src="" type="audio/mpeg" />
        Ваш браузер не підтримує аудіо.
    </audio>

    <div class="controls">
        <button id="btnPlay">▶ Відтворити</button>
        <button id="btnPause">⏸ Пауза</button>
        <button id="btnStop">■ Стоп</button>
        <button id="btnBack">⏮ -10с</button>
        <button id="btnFwd">⏭ +10с</button>
    </div>
    <div id="nowPlaying"></div>
</section>

<section class="list">
    <h3>Треки</h3>
    @if (!Model.Any())
    {
        <p>Додай свій перший трек вище </p>
    }
    else
    {
        <table class="tracks">
            <thead>
                <tr><th>Назва</th><th>Дія</th></tr>
            </thead>
            <tbody>
                @foreach (var t in Model)
                {
                    <tr data-url="@t.RelativeUrl" data-title="@t.Title">
                        <td>@t.Title</td>
                        <td class="actions">
                            <button class="playThis">Відтворити</button>
                            <form asp-controller="Tracks" asp-action="Delete" method="post" style="display:inline">
                                <input type="hidden" name="id" value="@t.Id" />
                                <button class="danger" type="submit">Видалити</button>
                            </form>
                        </td>
                    </tr>
                }
            </tbody>
        </table>
    }
</section>

```

Код-16 Views – Home – Index.cshtml

```

@using Microsoft.AspNetCore.Http
@inject IHttpContextAccessor HttpContext

<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="utf-8" />
    <title>Music Player</title>
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link rel="stylesheet" href="~/css/site.css" />
</head>
<body>
<header class="topbar">
    <div class="brand">🎵 Music Player</div>
    <nav>
        @if (HttpContext.User.Identity!.IsAuthenticated)
        {
            <form method="post" action="/Account/Logout" style="display:inline">
                <button class="link">Вийти з акаунту</button>
            </form>
            <form method="post" action="/System/Exit" style="display:inline;margin-left:1rem">
                <button class="danger">Вийти з програми</button>
            </form>
        }
    </nav>
</header>
<main class="container">
    @RenderBody()
</main>

<script src="~/js/player.js"></script>
</body>
</html>

```

Код-15 Views – Shared – _Layout.cshtml

```

body {
  font-family: system-ui, -apple-system, Segoe UI, Roboto, Arial, sans-serif;
  margin: 0;
}
.topbar {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 12px 16px;
  background: #111;
  color: #fff;
}
.topbar .brand {
  font-weight: 700;
}
.topbar .link,
.topbar .danger {
  background: none;
  border: none;
  color: #fff;
  cursor: pointer;
}
.topbar .danger {
  color: #ffdddd;
}
.container {
  max-width: 900px;
  margin: 24px auto;
  padding: 0 16px;
}
label {
  display: block;
  margin-top: 8px;
  font-size: 0.9rem;
}
input[type="text"],
input[name="username"],
input[name="password"] {
  width: 100%;
  padding: 8px;
  margin: 4px 0 10px;
}
button {
  padding: 8px 12px;
  border-radius: 8px;
  border: 1px solid #ccc;
  background: #f5f5f5;
  cursor: pointer;
}
button.danger {
  border-color: #dd6666;
  background: #ffe9e9;
}
.alert {
  background: #fff7d6;
  border: 1px solid #e6d28a;
  padding: 8px 12px;
  margin: 8px 0;
  border-radius: 8px;
}

```

Код-16.1 wwwroot – css – site.css

```

.uploader,
.player,
.list {
  margin-top: 20px;
}
.controls button {
  margin-right: 6px;
}
table.tracks {
  width: 100%;
  border-collapse: collapse;
  margin-top: 10px;
}
table.tracks th,
table.tracks td {
  border-bottom: 1px solid #eee;
  padding: 10px;
  text-align: left;
}
.actions {
  white-space: nowrap;
}
#nowPlaying {
  margin-top: 8px;
  color: #444;
  font-size: 0.95rem;
}

```

Код-16.2 wwwroot – css – site.css

```

src / js / player.js / ...
(function () {
  const audio = document.getElementById("player");
  if (!audio) return;

  const srcEl = document.getElementById("playerSource");
  const now = document.getElementById("nowPlaying");

  const btnPlay = document.getElementById("btnPlay");
  const btnPause = document.getElementById("btnPause");
  const btnStop = document.getElementById("btnStop");
  const btnBack = document.getElementById("btnBack");
  const btnFwd = document.getElementById("btnFwd");

  const rows = document.querySelectorAll("table.tracks tbody tr");
  rows.forEach((r) => {
    const btn = r.querySelector(".playThis");
    btn.addEventListener("click", () => {
      const url = r.getAttribute("data-url");
      const title = r.getAttribute("data-title");
      playUrl(url, title);
    });
  });

  function playUrl(url, title) {
    if (!url) return;
    srcEl.src = url;
    audio.load();
    audio.play();
    now.textContent = `Зараз грає: ${title}`;
  }

  btnPlay.addEventListener("click", () => audio.play());
  btnPause.addEventListener("click", () => audio.pause());
  btnStop.addEventListener("click", () => {
    audio.pause();
    audio.currentTime = 0;
  });
  btnBack.addEventListener("click", () => {
    audio.currentTime = Math.max(0, audio.currentTime - 10);
  });
  btnFwd.addEventListener("click", () => {
    audio.currentTime = Math.min(audio.duration || 0, audio.currentTime + 10);
  });

  if (rows.length > 0) {
    const first = rows[0];
    srcEl.src = first.getAttribute("data-url");
    now.textContent = `Обрано: ${first.getAttribute("data-title")}`;
  }
})();

```

Код-17 wwwroot – js – player.js

```
<Project Sdk="Microsoft.NET.Sdk.Web">
  <PropertyGroup>
    <TargetFramework>net9.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="9.0.10">
      <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
      <PrivateAssets>all</PrivateAssets>
    </PackageReference>
    <PackageReference Include="Pomelo.EntityFrameworkCore.MySql" Version="9.0.0" />
  </ItemGroup>
</Project>
```

Код-18 MusicPlayerWeb.csproj

```

Program.cs
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.StaticFiles;
using Microsoft.EntityFrameworkCore;
using MusicPlayerWeb.Data;
using MusicPlayerWeb.Services;

var builder = WebApplication.CreateBuilder(args);

var connectionString = builder.Configuration.GetConnectionString("DefaultConnection");
builder.Services.AddDbContext<AppDbContext>(options =>
    options.UseMySQL(connectionString, ServerVersion.AutoDetect(connectionString)));

builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
    .AddCookie(options =>
    {
        options.LoginPath = "/Account/Login";
        options.LogoutPath = "/Account/Logout";
        options.AccessDeniedPath = "/Account/Login";
    });

builder.Services.AddControllersWithViews();

builder.Services.AddScoped<IUserService, DbUserService>();
builder.Services.AddScoped<ITrackService, DbTrackService>();

builder.Services.AddHttpContextAccessor();
var app = builder.Build();

var provider = new FileExtensionContentTypeProvider();
provider.Mappings[".m4a"] = "audio/mp4";
provider.Mappings[".flac"] = "audio/flac";
app.UseStaticFiles(new StaticFileOptions { ContentTypeProvider = provider });

if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    app.UseHsts();
}
app.UseHttpsRedirection();
app.UseRouting();

app.UseAuthentication();
app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

var uploadsPath = Path.Combine(app.Environment.WebRootPath, "uploads");
Directory.CreateDirectory(uploadsPath);

app.Run();

```

Код-19 Program.cs

Висновок:

У ході роботи я навчилася створювати діаграми розгортання, компонентів і послідовностей для проєктованої системи. Отримала практичні навички моделювання структури та взаємодії елементів системи за допомогою UML-нотації.

Відповіді на контрольні запитання:

Відповіді на контрольні питання:

1. Що собою становить діаграма розгортання?

Діаграма розгортання (Deployment Diagram) показує фізичну архітектуру системи: як програмне забезпечення розміщується на апаратних вузлах, які зв'язки існують між вузлами, а також як компоненти системи розташовані на цих вузлах. Вона корисна для аналізу інфраструктури, серверів, пристроїв і мережевих з'єднань.

2. Які бувають види вузлів на діаграмі розгортання?

- Фізичні вузли (Node): апаратні пристрої або сервери.
- Програмні вузли: середовища виконання (наприклад, JVM, веб-сервер).
- Вузли баз даних: сервери баз даних або сховища.
- Компоненти всередині вузлів: програмні модулі або артефакти, що розміщені на фізичних вузлах.

3. Які бувають зв'язки на діаграмі розгортання?

- Зв'язок асоціації (Association): показує мережеве з'єднання між вузлами.
- Залежність (Dependency): вказує, що один вузол використовує інший.
- Комунікаційний канал (Communication path): фізичне з'єднання між вузлами для обміну даними.

4. Які елементи присутні на діаграмі компонентів?

- Компоненти (Component): незалежні модулі програмного забезпечення з чітко визначеними інтерфейсами.
- Інтерфейси (Interface): точки взаємодії компонентів.
- Пакети (Package): групування компонентів.
- Актори (Actor) або зовнішні системи: можуть бути присутніми для позначення джерел або отримувачів даних.

5. Що становлять собою зв'язки на діаграмі компонентів?

- Залежність (Dependency): один компонент залежить від іншого.
- Асоціація (Association): компоненти взаємодіють між собою.

- Реалізація інтерфейсу (Realization): компонент реалізує певний інтерфейс.
- Інформаційний потік (Information flow): показує рух даних між компонентами.

6. Які бувають види діаграм взаємодії?

- Діаграма послідовностей (Sequence Diagram).
- Діаграма комунікацій (Communication / Collaboration Diagram).
- Діаграма часових подій (Timing Diagram).
- Діаграма взаємодії (Interaction Overview Diagram).

7. Для чого призначена діаграма послідовностей?

Діаграма послідовностей описує динамічну взаємодію між об'єктами у вигляді послідовності повідомлень у часі. Вона використовується для аналізу логіки сценаріїв та послідовності виконання операцій у системі.

8. Які ключові елементи можуть бути на діаграмі послідовностей?

- Об'єкти / Лайнери життя (Lifeline): учасники взаємодії.
- Повідомлення (Message): обмін інформацією між об'єктами.
- Активації (Activation): період, коли об'єкт активний.
- Умовні гілки / Цикли (alt, loop, opt): відображення логічних конструкцій.
- Примітки (Note): додаткова інформація про процеси.

9. Як діаграми послідовностей пов'язані з діаграмами варіантів використання?

Діаграми послідовностей деталізують сценарії, які описані у діаграмах варіантів використання (UseCaseDiagram). Кожен варіант використання можна розкласти на кроки у вигляді повідомлень між об'єктами на діаграмі послідовностей.

10. Як діаграми послідовностей пов'язані з діаграмами класів?

Діаграми послідовностей показують об'єкти та їх взаємодію у часі, тоді як діаграми класів описують структуру цих об'єктів (класи, атрибути, методи). Іншими словами, діаграма класів визначає що існує, а діаграма послідовностей – як ці об'єкти взаємодіють.